Event Extraction Using Distant Supervision

Kevin Reschke[1], Martin Jankowiak[1], Mihai Surdeanu[2]
Christopher D. Manning[1], Daniel Jurafsky[1]
[1]Stanford University, 450 Serra Mall Stanford, CA 94305 USA
[2]University of Arizona, Gould-Simpson 811, 1040 E. 4th Street Tucson, AZ 85721 USA
kreschke@cs.stanford.edu, jankowiak@gmail.com, msurdeanu@email.arizona.edu
manning@cs.stanford.edu, jurafsky@stanford.edu

Abstract

Distant supervision is a successful paradigm that gathers training data for information extraction systems by automatically aligning vast databases of facts with text. Previous work has demonstrated its usefulness for the extraction of binary relations such as a person's employer or a film's director. Here, we extend the distant supervision approach to template-based event extraction, focusing on the extraction of passenger counts, aircraft types, and other facts concerning airplane crash events. We present a new publicly available dataset and event extraction task in the plane crash domain based on Wikipedia infoboxes and newswire text. Using this dataset, we conduct a preliminary evaluation of four distantly supervised extraction models which assign named entity mentions in text to entries in the event template. Our results indicate that joint inference over sequences of candidate entity mentions is beneficial. Furthermore, we demonstrate that the SEARN algorithm outperforms a linear-chain CRF and strong baselines with local inference.

Keywords: Distant-Supervision, Event-Extraction, Searn

1. Introduction

This paper explores a distant supervision approach to event extraction for knowledge-base population. In a distantly supervised setting, training texts are labeled automatically (and noisily) by leveraging an existing database of known facts. While this approach has been applied successfully to the extraction of binary relations such as a person's employer or a film's director (Mintz et al., 2009; Surdeanu et al., 2012), it has not previously been applied to event extraction.

We make three main contributions. First, we present a new research dataset for distantly supervised event extraction centered around airplane crash events. The dataset consists of a plane crash knowledge base derived from Wikipedia infoboxes and distantly generated entity-level labels covering a corpus of newswire text. Second, we use this dataset to conduct a preliminary evaluation of a number of extraction models. Our results serve as a baseline for further research in this domain. Third, our experiments demonstrate that joint learning (here using the SEARN algorithm (Daumé III, 2006)) performs better than several strong baselines, even in this complex and noisy setup.

2. Dataset and Slot-filling Task

We began by compiling a knowledge base of 193 plane crash infoboxes from Wikipedia's list of commercial aircraft accidents.[1] An example is shown in Table 1. From these we selected 80 single-aircraft crashes (40 for training; 40 for testing) that occurred after 1987. This is the timespan covered by our news corpus, which is comprised of Tipster-1, Tipster-2, Tipster-3, and Gigaword-5.[2]

We define a slot-filling task over eight slot types (⟨Flight Number⟩, ⟨Operator⟩, ⟨Aircraft Type⟩, ⟨Crash Site⟩, ⟨Passengers⟩, ⟨Crew⟩, ⟨Fatalities⟩, and ⟨Survivors⟩) as follows: Given a flight number, find values for the seven remaining slots. At test time, this involves retrieving relevant documents from our newswire corpus and assigning slot type labels (or NIL) to each entity in each document. For high recall, we retrieve any document containing the flight number string—e.g., *Flight 967*. This yielded 4,093 unique documents during training and testing.

Training data for these entity-level slot type decisions was generated by distant supervision. First, hand-crafted rules generated alias expansions for each slot value in the set of training events.[3] Then, for each training event, documents containing the flight number string and at least one slot value (or alias) were retrieved from the corpus. Named Entity Recognition software[4] was run on each document to identify entities, including numbers. Entity mentions matching a slot value (or alias) were marked as positive training examples for that slot type. Non-matching entities were marked as negative (NIL label) examples. NIL examples were subsampled to achieve a 50/50 split between positive and negative training examples. Table 2 shows frequencies for each label. We make these noisily generated training examples available as stand-off annotations.[5]

3. Experiments

Having introduced the general framework for distantly supervised event extraction, in this section we present experiments testing various models in the framework.

[1] http://en.wikipedia.org/wiki/List_of_accidents_and_incidents_involving_commercial_aircraft

[2] Available at catalog.ldc.upenn.edu/LDC93T3A and catalog.ldc.upenn.edu/LDC2011T07.

[3] E.g., *Airbus* is an alias for *Airbus A320-211*, and *eight* is an alias for *8*.

[4] http://nlp.stanford.edu/software/CRF-NER.shtml

[5] http://nlp.stanford.edu/projects/dist-sup-event-extraction.shtml

Slot Type	Slot Value
Wikipedia Title	Armavia Flight 967
isSinglePlaneCrash	true
Aircraft Name	N.A.
Aircraft Type	Airbus A320-211
Crash Date	3 May 2006
Crash Type	controlled flight into terrain pilot error
Crew	8
Fatalities	113
Flight Number	Flight 967
Injuries	0
Operator	Armavia
Passengers	105
Crash Site	Sochi International Airport, Black Sea
Survivors	0
Tail Number	EK-32009

Table 1: Sample plane crash infobox.

Label	Frequency	Named Entity Type
NIL	19196	
Crash Site	10365	LOCATION
Operator	4869	ORGANIZATION
Fatalities	2241	NUMBER
Aircraft Type	1028	ORGANIZATION
Crew	470	NUMBER
Survivors	143	NUMBER
Passengers	121	NUMBER
Injuries	0	NUMBER

Table 2: Label frequency in noisy training data.

3.1. Experiment 1: Simple Local Classifier

First we used multi-class logistic regression to train a model which classifies each mention independently, using the noisy training data described above. Features include the entity mention's part of speech, named entity type, surrounding unigrams, incoming and outgoing syntactic depencies, the location within the document and the mention string itself.[6] These features fall into five groups, detailed in Table 3. Each of the models described in this paper uses these five features sets.

We compare this local classifier to a majority class baseline. The majority baseline assigns the most common label for each named entity type as observed in the training documents (see Table 2). Concretely, all locations are labeled ⟨Site⟩, all organizations are labeled ⟨Operator⟩, all numbers are labeled ⟨Fatalities⟩, and all other named entities are labeled NIL. The remaining five labels are never assigned.

To compare performance on the final slot prediction task, we define precision and recall as follows. Precision is the number of correct guesses over the total number of guesses. Recall is the number of slots correctly filled over the number of findable slots. A slot is *findable* if its true value appears somewhere as a candidate mention. We do not penalize the extraction model for missing a slot that either was not in the corpus or did not occur under our heuristic notion of document relevance. For multi-valued slots, full recall credit is awarded if at least one value is correctly identified. For example, the slot-fill *Mississippi* would receive full credit for the crash site *Mississippi, U.S.A.*

NE	Named Entity Features: Unigrams and part-of-speech tags within the named entity mention, the number of tokens in the mention, and the named entity type of the mention.
LCon	Local Context: Unigrams and part-of-speech tags within five tokens of the named entity mention, with specific features for one, two, and three tokens before and after the mention.
SCon	Sentence Context: Unigrams and part-of-speech tags in the same sentence as the target named entity.
Dep	Dependency Features: Incoming and outgoing dependency arcs, lexicalized and unlexicalized.
LiD	Location in document: Is the target named entity mention in the first, second, third, or fourth quarter of the document?

Table 3: Feature sets for mention classification.

	Precision	Recall	F_1 Score
Maj. Baseline	0.026	0.237	0.047
Local Classifier	0.158	0.394	0.218

Table 4: Performance of local classifier vs. baseline.

Crash Site	8/50 = 0.16
Operator	5/25 = 0.20
Fatalities	7/35 = 0.20
Aircraft Type	4/19 = 0.21
Crew	15/170 = 0.09
Survivors	1/1 = 1.0
Passengers	11/42 = 0.26
Injuries	0/0 = NA

Table 5: Accuracy of local classifier by slot type

The performance of the local and majority classifiers are shown in Table 4. The test set contained 40 test infoboxes with a total of 135 findable slots. The local classifier considerably outperformed the baseline. Table 5 breaks down the accuracy of the local classifier by slot type.

3.2. Experiment 2: Sequence Model with Local Inference

The local model just presented fails to capture dependencies between mention labels. For example, ⟨Crew⟩ and ⟨Passenger⟩ go together; ⟨Site⟩ often follows ⟨Site⟩; and ⟨Fatalities⟩ never follows ⟨Fatalities⟩:

- **4** crew and **200** passengers were on board.
- The plane crash landed in **Beijing**, **China**.
- * **20** died and **30** were killed in last Wednesday's crash.

In this experiment, we compare our simple local model to a sequence model with local inference (SMLI). We implement SMLI using a maximum entropy markov model (MEMM) approach. In the local model, mentions in a sentence are classified independently. In contrast, at each step

[6] Parsing, POS tagging, and NER: Stanford Core NLP. nlp.stanford.edu/software/corenlp.shtml

	Precision	Recall	F_1 Score
Local Model	0.157	0.394	0.218
SMLI	0.153	0.417	0.224

Table 6: Performance of sequence model with local inference (SMLI).

in SMLI, the label of the previous non-NIL mention is used as a feature for the current mention. At training time, this is the previous non-NIL mention's noisy "gold" label. At test time, this is the classifier's output on the previous non-NIL mention.

Table 6 shows test-set results. SMLI boosted recall with only a slight decrease in precision. The difference in recall was statistically significant ($p < 0.05$).[7] Qualitative analysis of SMLI's feature weights revealed that the classifier learned the patterns mentioned above, as well as others.

3.3. Experiment 3: Noisy-OR Aggregation

So far we have assumed *exhaustive* label aggregation—as long as at least one mention of a particular value gets a particular slot label, we use that value in our final slot-filling decision. For example, if three mentions of *Mississippi* receive the labels ⟨Crash Site⟩, ⟨Operator⟩, and NIL, then the final slot-fills are Crash Site = *Mississippi* and Operator = *Mississippi*. Intuitively, this approach is suboptimal, especially in a noisy data environment where we are more likely to misclassify the occasional mention. In fact, a proper aggregation scheme can act as fortification against noise induced misclassifications.

With this in mind, we adopted *Noisy-OR* aggregation. The key idea is that classifiers give us distributions over labels, not just hard assignments. A simplified example is given below for two mentions of *Stockholm*.

- Stockholm ⟨NIL:0.8; Crash Site: 0.1, Crew:0.01, etc.⟩
- Stockholm ⟨Crash Site: 0.5; NIL: 0.3, Crew:0.1, etc.⟩

Given a distribution over labels ℓ for each mention m in M (the set of mentions for a particular candidate value), we can compute *Noisy-OR* for each label as follows.

$$Noisy\text{-}OR(\ell) = Pr(\ell|M) = 1 - \prod_{m \in M} (1 - Pr(\ell|m))$$

In the *Stockholm* example above, the *Noisy-OR* for ⟨Crash Site⟩ and ⟨Crew⟩ are 0.95 and 0.11 respectively. A value is accepted as a slot filler only if the *Noisy-OR* of the slot label is above a fixed threshold. We found 0.9 to be an optimal threshold by cross-validation on the training set.

Table 7 shows test-set results comparing *Noisy-OR* and *exhaustive* aggregation on the local and SMLI classifiers. We see that *Noisy-OR* improves precision while decreasing recall. This is expected because *Noisy-OR* is strictly more conservative (NIL-prefering) than *exhaustive* aggregation. In terms of F_1 score, *Noisy-OR* aggregation is significantly better at $p < 0.1$ for the local model and $p < 0.05$ for the SMLI.

		Precision	Recall	F_1 Score
Local	Exhaustive	0.158	0.394	0.218
	Noisy-OR	0.187	0.370	0.248
SMLI	Exhaustive	0.153	0.417	0.224
	Noisy-OR	0.185	0.386	0.250

Table 7: Exhaustive vs. Noisy-OR Aggregation.

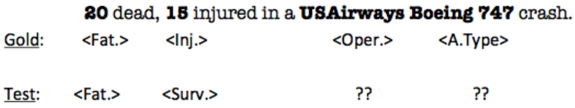

Figure 1: Error propagation in SMLI classification.

3.4. Experiment 4: Joint Models

In the previous two experiments, SMLI had better recall than our local model, but overall improvement was modest. One possible explanation comes from an error propagation problem endemic to this class of models. Consider the example in Figure 1. At training time, *USAirways* has the feature PREV-LABEL-INJURY. But suppose that at inference time, we mislabel *15* as ⟨Survivors⟩. Now *USAirways* has the feature PREV-LABEL-SURVIVOR, and we are in a feature space that we never saw in training. Thus we are liable to make the wrong classification for *USAirways*. And if we make the wrong decision there, then again we are in an unfamiliar feature space for *Boeing 747* which may lead to another incorrect decision.

This error propagation is particularly worrisome in our distant supervision setting due to the high amount of noise in the training data. To extend the example, suppose instead that at distant supervision time, *15* was given the incorrect "gold" label ⟨Fatalities⟩. Now at test time, we might correctly classify *15* as ⟨Injuries⟩, but this will put us in an unseen feature space for subsequent decisions because *USAirways* saw ⟨Fatalities⟩ at training time, not ⟨Injuries⟩.

An ideal solution to this error propagation problem should do two things. First, it should allow suboptimal local decisions that lead to optimal global decisions. For the previous example, this means that our choice for *15* should take into account our future performance on *USAirways* and *Boeing 747*. Second, models of sequence information should be based on actual classifier output, not gold labels. This way we are not in an unfamiliar feature space each time our decision differs from the gold label.

In essence, we want a joint mention model—one which optimizes an entire sequence of mentions jointly rather than one at a time. To this end, we tested two joint models: i) a linear-chain CRF[8], and ii) the SEARN algorithm (Daumé III, 2006). The following sections describe our implementation of these models and experimental results.

3.4.1. Linear-Chain CRF Model

Conditional random fields (CRFs) provide a natural language for joint modeling of sequences of mentions and their associated labels (Lafferty et al., 2001). CRFs are particularly well-suited to classification because they are discrim-

[7] All significance tests reported in this paper were computed using bootstrap resampling on test events with 10,000 trials.

[8] Implemented using Factorie (McCallum et al., 2009)

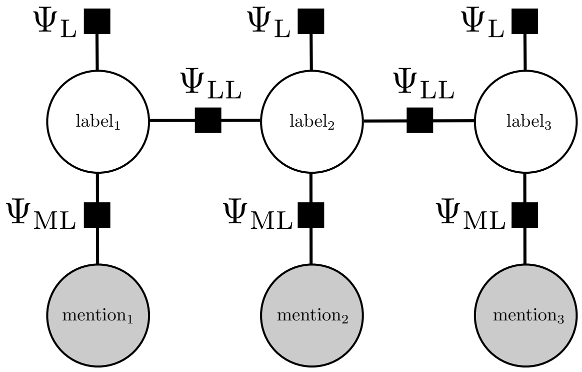

Figure 2: Linear-chain CRF for mention classification.

inative models, i.e. they do not involve modeling dependencies among mention features. For a sequence of mentions $\mathbf{m}$ and associated labels $\mathbf{l}$, the conditional probability is given as

$$P(\mathbf{l}|\mathbf{m}) = \frac{1}{Z(\mathbf{m})} \prod_{\Psi_i} \exp \left\{ \sum_{\mathrm{j}} \lambda_{\mathrm{ij}} \mathrm{f}_{\mathrm{ij}}(\mathbf{l}, \mathbf{m}) \right\} \qquad (1)$$

where we have introduced a set of factors $\{\Psi_i\}$, weights $\{\lambda_{ij}\}$, and features $\{f_{ij}\}$. We specialize to a linear-chain CRF with three factors: Ψ_{L}, Ψ_{LL}, and Ψ_{ML} (see Figure 2). The first factor captures label frequencies, the second captures dependencies between labels of adjacent mentions, and the third captures dependencies between labels and mention features.

Learning proceeds via stochastic gradient descent in conjunction with the max-product algorithm, which is also used during inference. Parameter updates are made using confidence weighting with a learning rate of unity. Hyperparameters are chosen by maximizing the F_1 score on a dev set, i.e. after *Noisy-OR* aggregation, which resulted in the following choices. Learning stops after $n_{\mathrm{T}} = 4$ rounds. In Eqn. NoisyOR, only the $n_{\mathrm{TOP}} = 3$ most probable mentions enter the product for any given label. Finally, the Ψ_{ML} weights corresponding to NIL were reduced by a multiplicative factor $x = 1.7$ to prevent too many NIL labels at inference.

3.4.2. SEARN Model

For our second joint model, we use the SEARN algorithm to infuse global decisions into a sequence tagger. SEARN is a general framework for training classifiers which make globally optimized choices in a structured prediction task (Daumé III, 2006). In our setting, SEARN generates a model in which a mention's label depends not only on its features and the previous non-NIL label, but also on the impact of this label for subsequent decisions later in the sentence.

The algorithm operates by associating training mentions with cost-vectors corresponding to the global, sequence-wide impact of different label choices. These mentions and cost-vectors are passed to a cost-sensitive classifier for learning. In our implementation, we follow Vlachos and Craven (2011) in using the cost-sensitive classifier described in Crammer et al. (2006), which amounts to a passive-aggressive multiclass perceptron.

Algorithm 1: SEARN as a sequence labeling algorithm.

```
input  : $T$ = training sentences,
         $\pi$ = initial gold label policy,
         $C$ = cost-sensitive classifier,
         $k$ = number of iterations
Initialize current hypothesis $h \leftarrow \pi$
for $k$ iterations do
    Initialize set of cost-sensitive examples $S \leftarrow \emptyset$
    for sentence $s$ in $T$ do
        for mention $m$ in $s$ do
            Classify mentions left of $m$ using $h$
            Compute features $\phi$ for $m$
            Initialize cost vector $c = \langle\rangle$
            for each possible label $l$ do
                Let $m$ have label $l$:
                Classify mentions right of $m$
                Let cost $c_m \leftarrow$ total errors in $s$
            end
            Add cost-sensitive example ($\phi$,$c$) to $S$
        end
    end
    Learn a classifier: $h' \leftarrow C(S)$
    Interpolate: $h \leftarrow \beta h' + (1 - \beta)h$
end
output: $h$ with $\pi$ removed
```

Inherent in this setup is the following chicken-and-egg problem: we want to train an optimal classifier based on a set of global costs, but we would like these global costs to be computed from the decisions made by an optimal classifier. SEARN gives an iterative solution to this problem. Algorithm 1 illustrates the basic framework. The algorithm is seeded with an initial policy based on gold labels (akin to our local sequence model, which uses gold labels for previous-label features during training). At each iteration, a new policy is learned from a cost-sensitive classifier and interpolated with previous policies.

SEARN has a number of hyperparameters. By cross-validation on the training set, we arrived at the following settings: 4 SEARN iterations; 8 perceptron epochs per iteration; interpolation β = 0.3; perceptron *aggressiveness* = 1.0.

3.4.3. Joint Model Results

The test-set results comparing these joint models to SMLI and our local model are shown in Table 8. All results use *Noisy-OR* aggregation. Our SEARN model outperformed the other models in precision and F_1 score ($p < 0.15$). The SEARN algorithm was able to model the inter-mention dependencies described in Section 3.2 while avoiding the error propagation problem affecting SMLI.

Our CRF model was able to learn useful weights for label pairs. For example, it learned a high positive weight for ⟨Passengers, Crew⟩ and a low negative weight for ⟨Fatalities, Fatalities⟩. However, performance did not improve over our non-joint models. One explanation for this

	Precision	Recall	F_1 Score
Local Model	0.287	0.370	0.248
Pipeline Model	0.185	0.386	0.250
CRF Model	0.159	0.425	0.232
SEARN Model	0.240	0.370	0.291

Table 8: Performance of joint, SMLI and local models.

	Precision	Recall	F_1 Score
LiD+Dep+Scon+LCon+NE	0.240	0.370	0.291
Dep+Scon+Lcon+NE	0.245	0.386	0.300
Scon+Lcon+NE	0.240	0.330	0.278
Lcon+NE	0.263	0.228	0.244
NE	0.066	0.063	0.064

Table 9: Feature ablation study on SEARN model.

comes from a key structural difference between our CRF model and our SEARN and Pipeline models. In our CRF model, edges connect adjacent named entities. In both SEARN and SMLI, the dependency is with the previous non-NIL named entity, ignoring any NIL labels that intervene. This means the latter two models are more directly sensitive to non-NIL labelings much earlier in the sentence. The lack of a non-NIL label early in a sentence turns out to be a strong signal that the sentence is not relevant to the planecrash domain. Without this signal, the CRF classifier frequently makes false-positive mislabelings in irrelevant sentences, e.g. assigning ⟨Site⟩ to a location not related to the crash. In general, the CRF model assigned labels more liberally than the other models, leading to high recall, but lower precision.

3.5. Experiment 5: Feature Ablation

In this final experiment, we conducted a features ablation study to explore the impact of different input features. Our models use five types of features as described in Table 3. Table 9 shows the performance of our SEARN model as feature sets are removed (without retuning hyperparameters). Performance actually increases as location in document (LiD) features are removed, but this result is not statistically significant. Removing dependency (Dep) features causes a significant drop in F_1 score ($p < 0.1$). Removing sentence context (SCon) features causes a less significant drop ($p = 0.16$). Finally, removing local context (LCon) features causes a major decrease in performance ($p < 0.01$).

4. Conclusion

This paper has presented a preliminary study of distant supervision applied to event extraction. We described a new publicly available dataset and extraction task based on plane crash events from Wikipedia infoboxes and newswire text. We presented five experiments. In the first experiment, we showed that a simple local classifier with a rich set of textual features outperforms a naive baseline, depite having access only to noisy, automatically generated training data. In the second experiment, we extended our approach to a sequence tagging model with local inference, showing that by considering previous label decisions as features, recall improves. In our third experiment, we demonstrated the effectiveness of a *Noisy-OR* model for label aggregation. In experiment four, we evaluated two models which apply joint inference to the sequence labeling task. Our linear-chain CRF model learned reasonable weights and improved recall, but overall performance suffered. Our second joing model, based on the SEARN algorithm, performed best, with considerable boost to both precision and F_1 score. Lastly, with a post-hoc ablation experiment, we showed that syntactic information and local context are both important for model success.

Acknowledgements

We gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Deep Exploration and Filtering of Text (DEFT) Program under Air Force Research Laboratory (AFRL) contract no. FA8750-13-2-0040. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

5. References

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585.

Hal Daumé III. 2006. *Practical Structured Learning Techniques for Natural Language Processing*. Ph.D. thesis, University of Southern California, Los Angeles, CA, August.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Andrew McCallum, Karl Schultz, and Sameer Singh. 2009. FACTORIE: Probabilistic programming via imperatively defined factor graphs. In *Neural Information Processing Systems (NIPS)*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Andreas Vlachos and Mark Craven. 2011. Search-based structured prediction applied to biomedical event extraction. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 49–57. Association for Computational Linguistics.